CHALMERS

Institutionen för data- och informationsteknik
Avdelningen för datorteknik

TENTAMEN

KURSNAMN	**Digital- och datorteknik**
PROGRAM:	**Data- och elektroingenjör** **Åk 1/ lp 1**
KURSBETECKNING	**LEU431**
EXAMINATOR	**Lars-Eric Arebrink**
TID FÖR TENTAMEN	**2010-10-18 kl 14.00 – 18.00**
HJÄLPMEDEL	**Av institutionen utgiven** **”Instruktionslista för FLEX-processorn” (INS1)** **Tabellverk eller miniräknare får ej användas.**
ANSV LÄRARE: **Besöker tentamen**	**Lars-Eric Arebrink, tel. 772 5718** **vid flera tillfällen**
ANSLAG AV RESULTAT	**Resultatlistor anslås senast 2010-11-04 på kursens hemsida. Granskning av rättning på institutionen 2010-11-04 och 2010-11-05 kl 12.30-13.00.**
ÖVRIG INFORM. **BETYGSGRÄNSER.** **SLUTBETYG**	**Tentamen omfattar totalt 60 poäng.** **Onödigt komplicerade lösningar kan ge poängavdrag.** **Svar på uppgifter skall motiveras.** **Betyg 3: 24 poäng** **Betyg 4: 36 poäng** **Betyg 5: 48 poäng** **För slutbetyg 3, 4 eller 5 på kursen fordras betyg 3, 4 eller 5 på tentamen och godkända laborationer.**

1. I uppgift a-d nedan används 5-bitars tal. X = 01010 och Y = 11101.

a) Visa med penna och papper hur räkneoperationen R = X+Y utförs med en 5-bitars ALU. **(1p)**

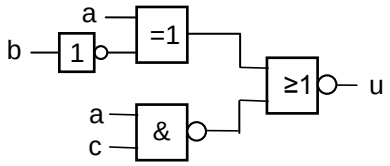
b) Vilka värden får flaggbitarna N, Z, V och C vid räkneoperationen? **(1p)**

c) Tolka bitmönstren R, X och Y som tal *utan* tecken och ange deras decimala motsvarighet. Vilken eller vilka flaggbitar visar om resultatet är korrekt vid tal utan tecken? **(1p)**

d) Tolka bitmönstren R, X och Y som tal *med* tecken och ange deras decimala motsvarighet. Vilken eller vilka flaggbitar anger om resultatet är korrekt vid tal med tecken? **(1p)**

e) Skriv det decimala talet 560,375 som ett 32-bitars flyttal enligt IEEE-standard 754-1985 (23 bitar av mantissan och 8 bitars karakteristika). Ge svaret på hexadecimal form. **(2p)**

f) Ge ett minimalt boolesk uttryck för u i grindnätet till höger.



(3p)

2. Det booleska uttrycket f(a,b,c) = (b + c)(a + b'+ c')(a'+ b + c') beskriver en boolesk funktion.

Konstruera ett grindnät med så få grindar som möjligt, som realiserar den booleska funktionen. NAND-grindar med valfritt antal ingångar, XOR-grindar och NOT-grindar får användas. Endast insignalerna a, b och c finns tillgängliga.

(4p)

3.

a) Ett synkront sekvensnät skall ha en insignal x och en utsignal u. Utsignalen u skall ges värdet "1" under ett bitintervall för varje insignalsekvens som består av ”exakt” två ettor följda av två nollor hos x. Med ”exakt” menas här att det inte får vara fler än två ettor.

Exempel: σ_x = ...0110011100110011001100111001...
σ_u = ...?000100000000100010001000000...

Utsignalen skall som i exemplet ges värdet "1" när den sista biten i en korrekt insignalsekvens anländer på x-ingången och behålla värdet "1" så länge x har kvar sitt värde under detta bitintervall.

Rita en tillståndsgraf för sekvensnätet. Hur många vippor skulle minst krävas vid realiseringen?

(3p+1p)

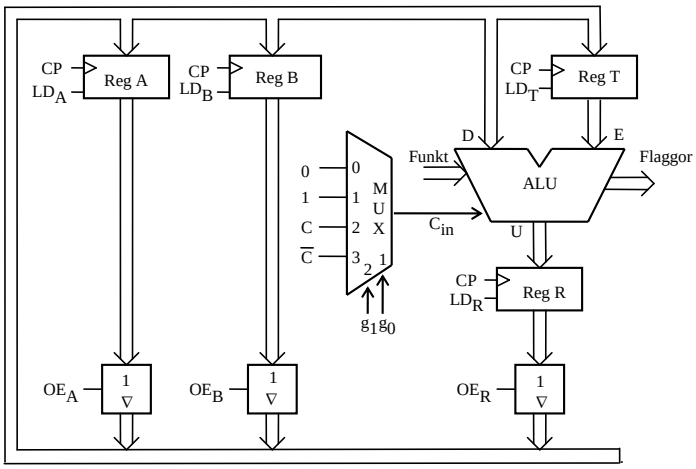
b) Realisera en autonom räknare med räknesekvensen $q_2q_1q_0$: 000, 011, 111, 110, 100, 001, 000. JK-vippor, NOR-grindar med valfritt antal ingångar och NOT-grindar får användas. **(6p)**

4. Ge RTN-beskrivning och styrsignaler för de tillstånd krävs för att utföra operationen enligt nedanstående RTN-beskrivning:

RTN-beskrivning: $7 \cdot A - 6 \cdot (B + 1) \rightarrow B$
(Aritmetisk multiplikation avses)

Använd den enkla datavägen till höger och ge ditt svar i tabellform.

Förutsätt att register A och B från början innehåller de data som skall behandlas enligt uttrycket ovan och att innehållen i register R och T är okända. Register A får inte ändras. Bortse från risken för overflow. Använd så få tillstånd som möjligt. Samtliga funktioner ALU:n kan utföra framgår av bilaga 1.



(5p)

5. Figur 1 i bilaga 3 visar hur datorn FLEX är uppbyggd. Bilaga 1 visar hur ALU'ns funktion väljs med styrsignalerna f_3 - f_0 och C_{in}.

I tabellen nedan visas RTN-beskrivningen av EXECUTE-sekvensen för en av FLEX-processorns instruktioner.

State	S-term	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_5	$Q_5 \cdot I_{xx}$	PC→MA, PC+1→PC	
Q_6	$Q_6 \cdot I_{xx}$	M→MA	
Q_7	$Q_7 \cdot I_{xx}$	2M→R, Flaggor → CC	
Q_8	$Q_8 \cdot I_{xx}$	R → M, NF	

NF i tabellens sista rad anger att nästa tillstånd (state) skall vara det första i FETCH-sekvensen.

a) Fyll i styrsignalkolumnen i tabellen och förklara vilken assemblerinstruktion som beskrivs. **(2p)**

b) Instruktionen nedan, som är ett villkorligt subrutinanrop, skall implementeras för FLEX-processorn med hjälp av styrenheten med fast logik. Operationskoden F0H skall användas.

BSREQ Adr RTN: If Z = 1: SP – 1 → SP
PC → M(SP)
PC + offset → PC;

else: PC + 1 → PC;

Gör en tabell liknande tabellen ovan för den efterfrågade EXECUTE-sekvensen. **(5p)**

6. Besvara kortfattat följande frågor rörande FLEX-processorn.

a) I instruktionsuppsättningen finns villkorliga hopp där hoppvillkoren avser tal med tecken. Då N-flaggan förekommer i dessa hoppvillkor är den alltid kombinerad med V-flaggan enligt N ⊕ V. Förklara varför! **(2p)**

b) Om man använder flera ORG-direktiv i ett program kan detta ställa till problem. Förklara hur! **(2p)**

c) Förklara varför det är användbart med en stack i minnet. När används stacken? Hur ”vet” processorn var i minnet stacken är placerad. **(3p)**

d) Översätt instruktionssekvensen till höger till maskinkod och visa hur den placeras i minnet. Det skall framgå hur offset för bransch-instruktionerna beräknas. **(3p)**

e) Hur lång tid tar instruktionssekvensen i d) att köra om FLEX-processorn klockas med frekvensen 1MHz? **(3p)**

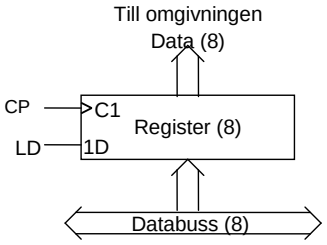
```
                ORG     $60
                LDX     #TAB
                LDAA    3,X
*
LOOP1           LDAB    4,X
*
LOOP2           DECB
                BNE     LOOP2
*
                INCA
                BMI     LOOP1
*
                BRA     NEXT
*
                RMB     4
TAB             FCB     0,-2,10,-4,20,-8,30

NEXT            NOP
```

7. FLEX-datorn som visas i bilaga 3 skall kompletteras med en utport på adressen FD_{16}. Principen för utporten visas i figuren till höger.

a) Konstruera ett grindnät som bildar signalen LD. Standardgrindar med valfritt antal ingångar får användas. **(3p)**

b) Adressen FD_{16} är gemensam för utporten och minnet. Vilka konsekvenser får detta? **(2p)**



c) Hur skriver man i programmet om man vill mata ut innehållet i register B till utporten? **(1p)**

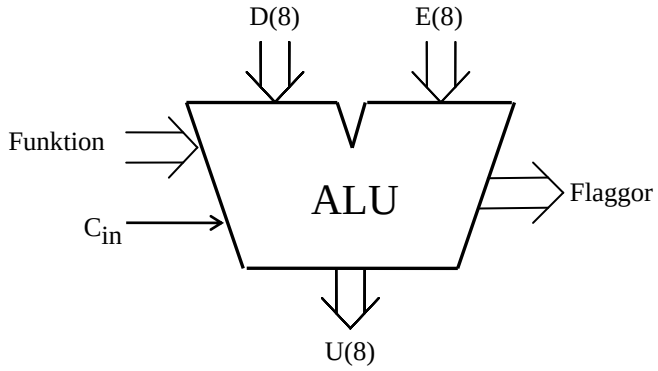
8. I minnet i ett datorsystem med FLEX-processorn finns en nollterminerad (= sista dataordet har värdet 0) sträng med sjubitars ASCII- tecken. Varje ASCII-tecken har en paritetsbit för jämn paritet som bit nr 7.

Skriv en subrutin i assemblerpråk för FLEX-processorn som tar reda på hur många av ASCII-tecknen i strängen som motsvarar hexadecimala siffror, dvs. 0-9 och A-F (stora bokstäver). Antalet sådana tecken skall finnas i B-registret vid återhopp. Vid anrop av subrutinen skall startadressen till strängen finnas i X-registret.

Endast register B och register CC får vara förändrade vid återhopp från subrutinen. För full poäng på uppgiften skall programmet vara korrekt radkommenterat. **(6p)**

Bilaga 1

ALU:ns funktion



ALU:ns **logik-** och **aritmetikoperationer** på indata **D** och **E** definieras av ingångarna **Funktion (F)** och $\mathbf{C_{in}}$ enligt tabellen nedan. **F** = **(f_3, f_2, f_1, f_0).**

I kolumnen Operation förklaras hur operationen utförs.

f_3 f_2 f_1 f_0	U = f(D,E,C_{in})	
	Operation	**Resultat**
0 0 0 0	bitvis nollställning	0
0 0 0 1		D
0 0 1 0		E
0 0 1 1	bitvis invertering	D_{1k}
0 1 0 0	bitvis invertering	E_{1k}
0 1 0 1	bitvis OR	D OR E
0 1 1 0	bitvis AND	D AND E
0 1 1 1	bitvis XOR	D XOR E
1 0 0 0	D + 0 + C_{in}	D + C_{in}
1 0 0 1	D + FFH + C_{in}	D – 1 + C_{in}
1 0 1 0		D + E + C_{in}
1 0 1 1	D + D + C_{in}	2D + C_{in}
1 1 0 0	D +E_{1k} + C_{in}	D – E – 1 + C_{in}
1 1 0 1	bitvis nollställning	0
1 1 1 0	bitvis nollställning	0
1 1 1 1	bitvis ettställning	FFH

Carryflaggan (C) innehåller minnessiffran ut (carry-out) från den mest signifikanta bitpositionen (längst till vänster) om en aritmetisk operation utförs av ALU:n.

Vid **subtraktion** gäller för denna ALU att
C = 1 om lånesiffra (borrow) uppstår och
C = 0 om lånesiffra inte uppstår.

Carryflaggans värde är 0 vid andra operationer än aritmetiska.

Overflowflaggan (V) visar om en aritmetisk operation ger "overflow" enligt reglerna för 2-komplementaritmetik.

V-flaggans värde är 0 vid andra operationer än aritmetiska.

Zeroflaggan (Z) visar om en ALU-operation ger värdet noll som resultat på U-utgången.

Signflaggan (N) är identisk med den mest signifikanta biten (teckenbiten) av utsignalen U från ALU:n.

Half-carryflaggan (H) är minnessiffran (carry) mellan de fyra minst signifikanta och de fyra mest signifikanta bitarna i ALU:n.

H-flaggans värde är 0 vid andra operationer än aritmetiska.

I tabellen ovan avser "**+**" och "**–**" **aritmetiska operationer**. Med t ex $\mathbf{D_{1k}}$ menas att samtliga bitar i **D** inverteras.

Bilaga 2

Assemblerspråket för FLEX-processorn.

Assemblerspråket använder sig av mnemoniska beteckningar liknande dem som processorkonstruktören MOTOROLA (FREESCALE) specificerat för maskininstruktioner för mikroprocessorerna 68XX och instruktioner till assemblatorn, s k pseudoinstruktioner eller assemblatordirektiv. Pseudoinstruktionerna listas i tabell 1.

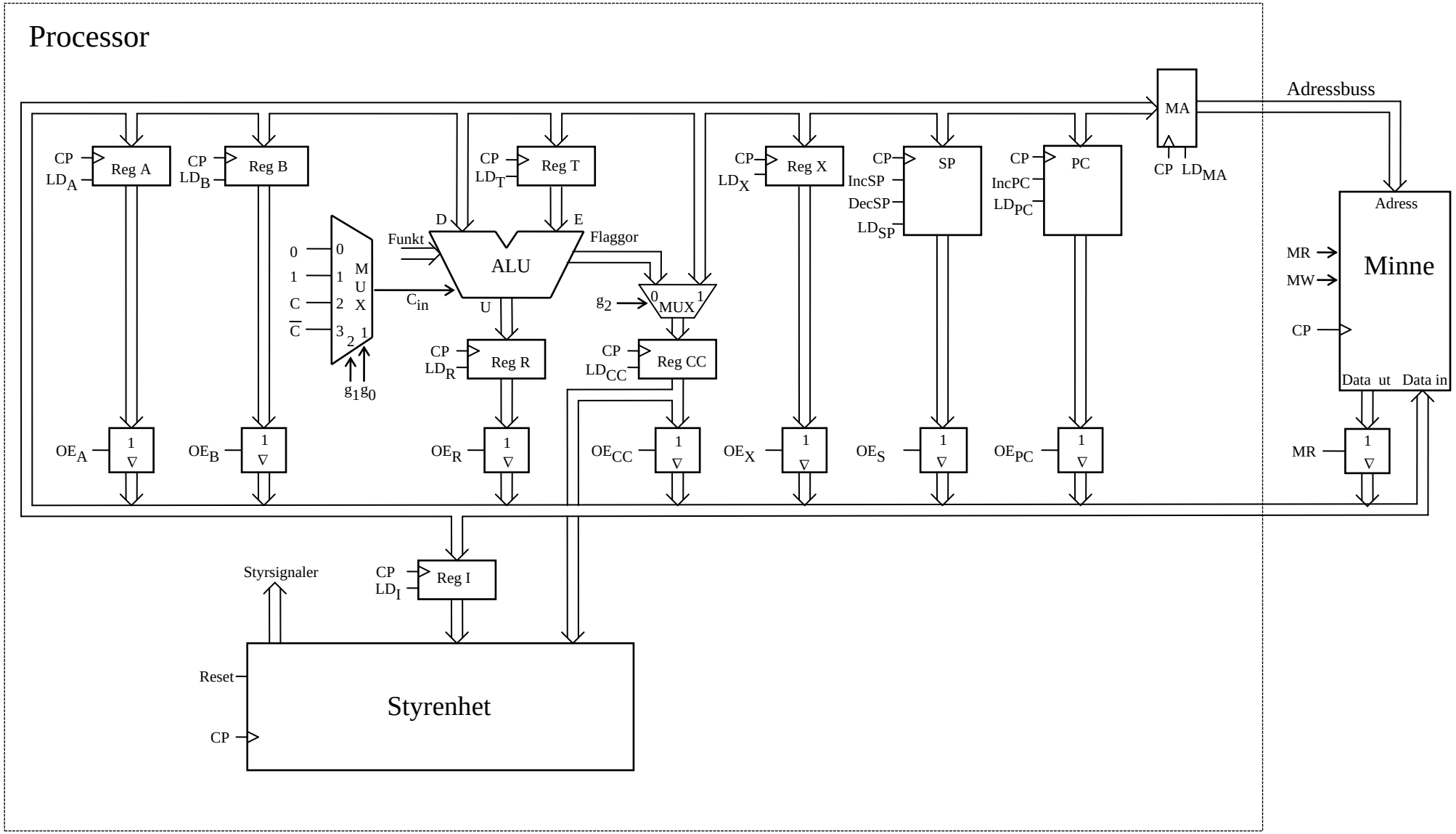
Tabell 1

Direktiv	Förklaring
ORG N	Placerar den efterföljande koden med början på adress N. (ORG för ORiGin = ursprung)
L RMB N	Avsätter N bytes i följd i minnet (utan att ge dem värden), så att programmet kan använda dem. Följden placeras med början på adressen L. (RMB för Reseve Memory Bytes)
L EQU N	Ger symbolen L konstantvärdet N. (EQU för EQUates = beräknas till)
L FCB N1,N2	Avsätter en byte för varje argument i följd i minnet. Respektive byte ges konstantvärdet N1, N2 etc. Följden placeras med början på adressen L. (FCB för Form Constant Byte)
L FCS ”ABC”	Avsätter en byte för varje tecken i teckensträngen ”ABC” i följd i minnet. Respektive byte ges ASCII-värdet för A B C, etc. Följden placeras med början på adressen L. (FCS för Form Character String)

Tabell 2 7-bitars ASCII

0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1	$b_6b_5b_4$ $b_3b_2b_1b_0$
NUL	DLE	SP	0	@	P	`	p	0 0 0 0
SOH	DC1	!	1	A	Q	a	q	0 0 0 1
STX	DC2	“	2	B	R	b	r	0 0 1 0
ETX	DC3	#	3	C	S	c	s	0 0 1 1
EOT	DC4	$	4	D	T	d	t	0 1 0 0
ENQ	NAK	%	5	E	U	e	u	0 1 0 1
ACK	SYN	&	6	F	V	f	v	0 1 1 0
BEL	ETB	‘	7	G	W	g	w	0 1 1 1
BS	CAN	(	8	H	X	h	x	1 0 0 0
HT	EM	)	9	I	Y	i	y	1 0 0 1
LF	SUB	*	:	J	Z	j	z	1 0 1 0
VT	ESC	+	;	K	[Ä	k	{ä	1 0 1 1
FF	FS	,	<	L	\Ö	l	|ö	1 1 0 0
CR	GS	-	=	M	]Å	m	}å	1 1 0 1
S0	RS	.	>	N	^	n	~	1 1 1 0
S1	US	/	?	O	_	o	RUBOUT (DEL)	1 1 1 1

Bilaga 4



Figur 1. Datorn FLEX.